Chapter 2
Introduction to CSS / CSS-1

CSS (Cascading Style Sheets)
- They are actually seperate files that we include into our HTML files.

- The General Rule:
that every line we write will follow...

```
selector {
  property: value;
  anotherProperty: value;
}
```

- 'Inline' styling is not a good idea because it doesn't seperate HTML and CSS files. Furthermore it's a lot of work to form lots of things all one at a time and especially when we want to make some changes to multiple elements,

```
<h3 style="color: pink;">My Heading</h3>
```

- 'Style Tag' is also a bad idea, -on the other hand generally used for quick demonstrations and checks- coz it's also included in the HTML file but as a seperate part on top in 'head' part; which is mainly allows us to write some CSS code inside this tag,

```
<style type="text/css">
  .
</style>
```

PS: / Comment in CSS */*

- So finally to write our CSS in a different file, we use <link> tag inside the HTML file. Hitting tab after typing 'link' will create a nicely prepared tag for us.,

```
<link rel="stylesheet" type="text/css" href="CSSFileName.css">
```

PS: It must be included in the 'head' section!

- There are three types of color system:
1.Hexadecimal
2.RGB
3.RGBA
Actually the former two refer to the same thing but have differences in both syntax and base selection. The differencelatter two

- Hexadecimal colours contain six digits following a '#'. It's using base 16.
`#_ _ _ _ _ _` : First 2 represents how much red is the colour and the others green, blue respectively.
red grn blu

```
#FFFFFF : White
#000000 : Black
```

- RGB colours have slight different showing style. It's using base 10, the values show red, green, blue

```
rgb(_, _, _)
     r  g  b
rgb(255,255,255): White
rgb(0,0,0): Black
```

- RGBA system adds a fouth field as transparency value whose range is between 0 and 1, (A stands for Alpha)

```
rgba(_, _, _, ._)
      r   g  b   t
```

`rgba(100,82,12,1)`: No transparency
`rgba(100,82,12,0)`: Fully transparent (nothing shows up)
`rgba(100,82,12,.6)`: Little transparency (as getting closer to 1)
`rgba(100,82,12,.3)`: More transparency (as getting closer to 0)

- Background Property Usage:

```
background: rgb(255,255,255); /* full color */
background:
url(https://badasshelmetstore.com/wp-content/uploads/2016/10/avengers-
helmets.jpg); /* image */
```

Alone this will replicate the image all along the page so be careful to pick a continous one if using this.

- Some Background Properties:

```
background-repeat: no-repeat; /* If we don't want the image to be repeated as above we use this
*/
background-size: cover; /* If we don't use this property the page won't cover the page. It won't
strecth out or won't fit depending on the resolution; if we don't use this */
```

- Border Usage:

Normally we use border with 3 must-have properties:
1. width
2. style
3. color

So the original code looks like this:

```
border-width: 3px;
border-style: solid;
border-color: white;
```

But we use the following in terms of being practical:

```
border: 3px solid white; /* widht style color respectively */
```

CSS Selectors:
1.Element
2.ID
3.Class

1.Element Selector:

Selects all instances of a given element.

```
div {
  background: aquamarine;
}
```

2.ID

Selects an element with a given ID. IDs have to be unique! It's not possible to have multiple instances with the same ID! We use hashtag '#' while referring to IDs.

```
<p id="mytext">Hallo!</p>
-------------------------
#mytext {
  background-color: yellow;
}
```

3.Class

Selects all elements with the given class. We use dot '.' while referring to classes.

```
<p class="important">Achtung!</p>
<p class="important">Das ist verboten!</p>
-----------------------------------------
.important {
  color: red;
}
```

- To load the checkboxes as checked while loading the page, we add 'checked' ATR in the input tag of the checkbox,

```
<input type="checkbox" checked>
Fertig!
```

- Google Chrome Inspect Element:

We can right click while on a webpage to inspect element. It will allow us to see both the HTML and the CSS part of it. So we can use this feature both for testing and trying things easily while creating some things and for inspecting stylings of other webpages and replicating them.

If we right click on a specific element and then click on inspect element it will directly show us that specific element within the opened up code segment.

Here on the right side, 'Styles' tab will consist the CSS stylings! We can directly on and off things or manipulate things here and watch the changes.

On the left side we can click on 'Magnifier' icon to go to sources of some things on the page by then clicking on'em.

More Selectors:

*/*Element*/* -> Selects all elements of the defined type

```
li {

}
```

*/*class*/* -> Selects all elements with the specified class

```
.hello {

}
```

*/*id*/* -> Selects an element with a specific and unique ID

```
#name {

}
```

*/*Star*/* -> Selects everything on a page

```
* {
   border: 1px solid lightgrey;
}
```

*/*Descendant Selector*/* -> Selects element(s) inside an element(s)

```
ul li a {
   color: red;
}
```

*/*Adjacent Selector*/* -> Selects an element after an element.

```
h4 + ul {
   border: 4px solid red;
}
```

*/*Attribute Selector*/* -> Selects all of that ATRs

```
a[href="http://www.google.com"] {
   background: blue;
}
```

*/*nth of type*/* -> Selects nth element of every odd/even number of that element

```
li:nth-of-type(3){
   background: purple;
}

li:nth-of-type(odd){
   background: purple;
}
```

Inheritence & Specificity

'Inheritence' is the traditional thing that we know from almost all languages, when we specify,

```
body {
   color: red;
}
```

Everything on the page, inherits this and turns into blue although we don't mark'em; if we want them to be styled differently we gotta write'em explicitly, like,

```
body {
   color: red;
}
ul {
   color: blue;
}
li:nth-of-type(3){
   color: green;
}
```

And here 'Specifity' moves in. It is multiple stylings targeting one element; just like above body, ul and li targets the 3rd li(s) on the page. And in this war, the closest one (in terms of containing) -the most specific one in other words- wins this war.

At this point when we inspect element on a page and look at the styles, we will see that those defeated inherited styles are overlined! And these are showed from bottom to top as more general to more specific one.

Specificity Order: From least to most,

1. */*Type Selectors*/*

```
li {

}

li a {

}

h4 + ul {

}
```

2. */*Class, Attribute & Pseudo-Class Selectors*/*

```
.hallo {

}
input[type="text"] {

}
a[href="http://www.google.com"] {

}
a:hover
input:checked
```

3. */*ID Selectors*/*

```
#hello {

}
```

MORE SPECIFIC SELECTORS!

*/*Make all "checked" checkboxes have a left margin of 50px(margin-left: 50px)*/*

```
input:checked {
   margin-left: 50px;
}
```

/ Make the <label> elements all UPPERCASE without changing the HTML(definitely look this one up*/*

```
label {
   text-transform: uppercase;
}
```

Other Options: lowercase, capitalize...

*/*Make the first letter of the element with id 'special' green and 100px font size(font-size: 100)*/*

```
#special::first-letter {
   color: green;
   font-size: 100px;
}
```

*/*Make the <h1> element's color change to blue when hovered over */*

```
h1:hover {
   color: blue;
}
```

*/*Make the <a> element's that have been visited gray */*

```
a:visited {
   color: gray;
}
```

Other Options: link, visited, hover, active...